(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT) **AB**

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
8 March 2001 (08.03.2001)

PCT

(10) International Publication Number
WO 01/17152 A1

(51) International Patent Classification[7]: H04K 1/04

(21) International Application Number: PCT/BR99/00076

(22) International Filing Date:
13 September 1999 (13.09.1999)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
PI 9903609-6 27 August 1999 (27.08.1999) BR

(71) Applicant *(for all designated States except US)*: **COPPE/UFRJ - COORDENACÃO DOS PROGRAMAS DE PÓS GRADUACÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO** [BR/BR]; Centro de Tecnologia, S/Nº, Bloco G, Ilha do Fundão, CEP-20000-000 Rio de Janeiro, RJ (BR).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: **CARDOSO SALOMÃO, Sérgio, Luiz** [BR/BR]; Rua Torres Sobrinho, 56, B1II apto, 204, Méier, CEP-20780-050 Rio de Janeiro, RJ (BR). **CASTRO ALVES, Vladimir** [BR/BR]; Rua 09, Quadra 87, Lote 5A, casa 12, Camboinhas, Niterói, CEP-24358-000 Rio de Janeiro, RJ (BR). **MONTEIRO CHAVES, Filho, Eliseu** [BR/BR]; Av. Mal. Henrique Lott, 270, apto 1104, Barra da Tijuca, CEP-22631-370 Rio de Janeiro, RJ (BR).

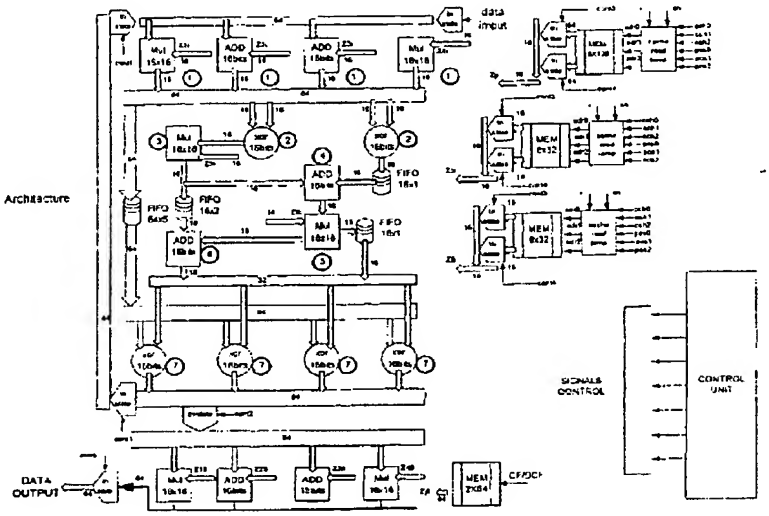
(74) Agents: **JOUBERT GONCALVES DE CASTRO & ZULDECH ASSESSORIA EMPRESARIAL LTDA**; Av. Ernani do Amaral Peixoto, 458 s1001, Centro Niterói, CEP-24020-077 Rio de Janeiro, RJ et al. (BR).

(81) Designated States *(national)*: AL, AM, AT, AU, AZ, BA, BB, BG, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GE, GH, GM, HR, HU, ID, IL, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, UA, UG, US, UZ, VN, YU, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, SD, SL, SZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

[Continued on next page]

(54) Title: A METHOD FOR THE HARDWARE IMPLEMENTATION OF THE IDEA CRYPTOGRAPHIC ALGORITHM - HIPCRYPTO



(57) Abstract: HiPCrypto is the hardware implementation of the most secure private key cryptographic algorithm: IDEA (International Data Encryption Algorithm), through the exploitation of spatial and temporal parallelism techniques in order to achieve the processing speed required by ATM and Gigabit networks.

WO 01/17152 A1

Published:
— *With international search report.*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

Description of the Invention "A METHOD FOR THE HARDWARE IMPLEMENTATION OF THE IDEA CRYPTOGRAPHIC ALGORITHM - HiPCrypto"

TECHNICAL FIELD

HIPCrypto is a hardware architecture proposal for the IDEA cryptographic algorithm, in which were used techniques for the exploitation of spatial and temporal parallelism, in order to reach the processing speeds required by real time applications and high speed data communication networks such as ATM.

Nowadays, a world tendency exists for the use of networks that provide different types of Telecommunication services such as the Integrated Service Data Network (ISDN). These types of networks should provide a wide range of services from telephone and cable TV to video conference.

The technological progress of transmission data networks pushed the development of cryptographic algorithm that became progressively more complex and robust. They are widely used by private and governmental organizations as well as individuals that need to ensure secrecy in data communication.

The increasing complexity of recent cryptographic algorithms require high processing capabilities due to the large number of arithmetic and logic operations that have to be executed, in some cases for real time applications like in video confereces.

PREVIOUS TECHNIQUES

Direct hardware implementation of cryptographic algorithms can ensure high processing speeds required by current and future applications in data

transmission and eliminate a potential bottleneck in data communication networks that require high security levels.

Consequently, several cryptographic algorithms were totally or partially implemented as Application Specific Integrated Circuits.

Several hardware and software implementations have been developed in the past decade for the Data Encryption Standard (DES), the most popular private key cryptographic algorithm. Table 1 shows the performance obtained for some software implementations in different platforms. Table 2 shows the performance obtained for some dedicated hardware implementations. From table 2 one can see that the 6868 integrated circuit from VLSI Technology reaches up to 512Mbit/s, which is not sufficient to support some high end ATM applications. Futhermore, it cryptanalysis on DES proved that it is weaker than some recent private key cryptographic algorithms like IDEA. Few hardware implementations of IDEA or its predecessors were reported in the litterature. For example, an ASIC that implements the PES algorithm, which originated IDEA, has reached up to 55 Mbits/s at 25 MHz.

DETAILED DESCRIPTION

IDEA cryptographic algorithm

The first form of the IDEA algorithm, was created by " Xuejia Lai and James Massey " in 1990 (US05214703 patent) and was called PES (Proposed Encryption Standard). In 1991, the algorithm was strengthened and was called IPES (Improved Proposed Standard Encryption). In 1992 IPES was called IDEA (International Data Encryption Algorithm), and is actually considered by many specialists

in the field of cryptography as the strongest existing symmetrical algorithm.

IDEA is a symmetric, block-oriented cryptographic algorithm, which uses 128-bit keys (thus making it practically immune to brute-force attacks) and 64-bit plaintext blocks. IDEA is build upon a basic function, which is iterated multiple times. As shown in Figure 1 the basic function is iterated eight times. The first iteration operates on the input 64-bit plaintext block and the successive iterations operate on the 64-bit block obtained from the previous iteration. After the last iteration, a final transformation step produces the 64-bit ciphertext block.

Figure 1 shows the structure of the basic function. It involves three simple operations: bitwise exclusive-or, addition modulo 2^{16} (addition, ignoring the " overflow ") and multiplication modulo $2^{16} + 1$ (multiplication, ignoring the " overflow "). For each iteration, the 64-bit input block is divided into four 16-bitsub-blocks. In Figure 1, X1, X2, X3 and X4 denote the four 16-bit input sub-blocks used by the each iteration. The 64-bit block produced by each iteration is also constituted by four 16-bit sub-blocks. In Figure 1, Y1(i), Y2(i), Y3(i) and Y4(i) denote the four sub-blocks resulting from the each iteration. The 128-bit key is divided into 52 16-bit sub-keys (sub-key generation is discussed ahead). Six sub-keys are used in each iteration and four sub-keys are used in the final transformation. In Figure 1, Z1(i), Z2(i), Z3(i), Z4(i), Z5(i) and Z6(i) denote the six sub-keys used in each iteration. The operations performed in the each iteration are:

1. Multiply sub-block X1(i) by sub-key Z1(i)
2. Add sub-block X2(i) and sub-key Z2(i)
3. Add sub-block X3(i) and sub-key Z3(i)
4. Multiply sub-block X4(i) by sub-key Z4(i)
5. XOR the results of (1) and (3)
6. XOR the results of (2) and (4)
7. Multiply the result of (5) by sub-key Z5(i)
8. Add the results of (6) and (7)
9. Multiply the result of (8) by sub-key Z6(i)
10. Add the results of (7) and (9)
11. XOR the results of (1) and (9)
12. XOR the results of (3) and (9)
13. XOR the results of (2) and (10)
14. XOR the results of (4) and (10)

The outputs of the iteration are the four sub-blocks produced by steps (11) to (14). The two inner sub-blocks from steps (12) and (13), Y2(i) to Y3(i), are swapped, except for the last iteration.

Figure 1 shows the structure of the final transformation. In this figure, Z1(9)to Z4(9) denote the four 16-bit sub-keys and Y1 to Y4 denote the four 16-bit sub-blocks of the 64-bit ciphertext block. The operations performed in the final transformation are:

15. Multiply sub-block X1 by sub-key Z1(9) to obtain Y1
16. Add sub-block X2 and sub-key Z2(9) to obtain Y2
17. Add sub-block X3 and sub-key Z3(9) to obtain Y3
18. Multiply sub-block X4 by sub-key Z4(9) to obtain Y4

The encryption and decryption sub-keys are generated from the single 128-bit key. Encryption sub-keys are generated as follows. Initially, the 128-bit key is

divided into eight 16-bit sub-keys. Six of these sub-keys, Z1(1) to Z6(1), are used in the first iteration. The two remaining sub-keys, Z1(2) and Z2(2), are for the second iteration. The original 128-bit key is then rotated left by 25 bits and the resulting key is again divided into eight 16-bit sub-keys. Four sub-keys, Z3(2) to Z6(2), are grouped with Z1(2) and Z2(2) and destined to the second iteration. The other four sub-keys, Z1(3) to Z4(3), are to be used in the third iteration. Next, the key is again rotated left by 25 bits, divided into eight 16-bit sub-keys and these sub-keys are grouped properly. This process is repeated each of the sub-keys for the eight iterations and for the final transformation have been generated. Decryption sub-keys are calculated as either the additive or the multiplicative inverses of the encryption keys.

As stated, the main goal in designing HiPCrypto is to obtain a device which would meet the performance requirements of applications in current and future high-speed data networks. This was achieved by including parallel execution techniques into the design of HiPCrypto's architecture. There are two opportunities for exploiting parallelism in the IDEA algorithm: in the execution of its basic function and in the iterations of this function.

Examining the data flow shown in Figure 1, one can identify groups of operations that are data independent. In each group, one operation does not use the results produced by other operations in the group. The sets of independent operations are: the multiply and add operations in steps (1) to (4); the exclusive-or operations in steps (5) and (6); and the exclusive-or operations in

steps (11) to (14). These independent operations can be performed simultaneously, provided the architecture incorporates multiple functional units dedicated to the execution of each of them.

By including multiple functional units in the architecture, we are making use of spatial parallelism. Temporal parallelism can also be employed in the execution of the basic function, by overlapping in time the operations upon distinct plaintext blocks. In this way, multiple blocks can be encrypted (or decrypted) simultaneously, instead of sequentially. This temporal parallelism was implemented with the pipeline shown in Figure 2.

Stage 1 contains two add and two multiply units that perform in parallel the independent operations in steps (1) to (4) of the algorithm.

Stage 2 contains two exclusive-or units to execute the operations in steps (5) and (6) in parallel.

Stages 3, 4, 5 and 6 contain a single add or multiply unit and they execute, respectively, the operations in steps (7), (8), (9) and (10) of the algorithm.

Stage 7 has four exclusive-or units to execute steps (11) to (14) in parallel.

The last stage has two add units and two multiply units and performs the algorithm's final transformation (see Figure 2). This stage will be referred to as the output stage.

In Figure 2, one can notice the inclusion of between stages of the pipeline. These queues temporarily hold data forwarded between non-adjacent stages. For

instance, stage 7 operates on sub-blocks from stages 1 and 5 (see Figure 1).

A sub-block from stage 1 arrives five cycles before the corresponding sub-block from stage 5, and during this time interval it remains in one of the queues connecting stages 1 and 7. When the sub-block from stage 5 is available, the sub-block in the front of the queue is dequeued and paired with the sub-block from stage 5. A queue is needed along the shortest path (in number of stages) between two non-neighbor stages. The size of each queue is indicated in figure 2.

The final aspect in HiPCrypto's architecture concerns the generation and storage of the sub-keys. To generate the encryption sub-keys, it would be necessary a circuitry for the rotation and sub-division of the 128-key. Moreover, the generation of the decryption sub-keys would require an arithmetic unit for the calculation of additive and multiplicative inverses. The inclusion of this additional hardware would only be reasonable if the key changes very frequently, say, every few blocks. But that is not the common case in a private-key cryptosystem: typically, the key shared by a group of partners is changed in a long term basis (days or weeks, for example). For this reason, only sub-key storage is provided. Sub-keys are generated externally by the host system, and then downloaded into the chip.

Architecture of HIPCrypto

The HIPCrypto architecture, Figure 3, executes a complete iteration of the algorithm. This architecture is composed of six 16-bit multipliers, six

16-bit adders and six 16-bit exclusive-or, memories for sub-key storage, buffers, tri-states and a control unit.

The operations contained in each stage of the pipeline, will be executed in an only machine cycle and since there are 7 pipeline stages, it will cipher (resp. decipher) 7 64 bits blocks for each execution of the algorithm.

The HIPCrypto was designed to offer four kinds of configurations, ie, 1, 2, 4 or 8 integrates in series (table 3).

Each pipeline segment is executed in one clock cycle. For one chip configuration, seven 64 bits blocks are processed each 56 (7 x 8) machine cycles. For 2 chips seven 64 bits blocks are processed each 28 (7 x 4) machine. For 4 chips configuration seven 64 bits blocks are processed each 14 (7 x 2) machine cycles. For 8 chips configuration seven 64 bits blocks are processed each 7 (7 x 1) machine cycles, that is to say, one 64 bits block for each machine cycle.

The proposed HIPCrypto's structure can be adapted to different uses. The adequate compromise between throughput and cost can be obtained by selecting the number of chips operating in series.

The signals used for selecting the chip configurations were divided in two groups: three signals that will define the configuration cch <2:0> and three signals that will define the position of the chip into the chain pos <2:0>. Tables 4 and 5 show respectively the configurations and the possible positions.

The sub-keys are stored in 4 RAMs according to Figures 3 and4.

For sub-keys Z1(i), Z2(i), Z3(i) and Z4(i), a 128 bits x 8 memory is used. The first 64 bits of each memory position, least significant bits, store the cipher sub-keys (positions 0 to 63) and the last 64 bits, most significant bits (positions 64 to 127), store the decipher sub-keys. The selection to execute the algorithm in cipher or decipher mode is made through the bus selection (see Figures 3 and 4).

For sub-keys Z5(i) and Z6(i), two 32 bits x 8 RAMs are used, where the 16 least significant bits (0 to 15), store the cipher sub-keys Z5(i) and Z6(i), and the 16 bits most significant store the decipher sub-keys.

For the sub-keys Z1(9), Z2(9), Z3(9) and Z4(9), a 64 bits x 2 memory is used.

Control Unit

The control unit (see Figure 3) is the operational block that controls the operation of the architecture. This unit together with some extra circuits is responsible for the generation of the control signals. The main functions of this unit are described in the following.

The control unit selects ciphering and deciphering modes, i.e. sleceting the cipher and decipher sub-keys respectively in each embedded memory.

The control unit also allows the correct initialization, feeding and synchronization of the pipeline stages by generating all enables and reset signals for each internal block.

The output stage will only be used by the last chip in each configuration. This selection is also

performed by the control unit through the selected configuration for each chip.

HIPCrypto performance

Table 7 shows some examples of the performance of HIPCrypto implemented in a two metal layer 0,7 micron CMOS technology.

CLAIMS

1. A METHOD FOR THE HARDWARE IMPLEMENTATION OF THE IDEA CRYPTOGRAPHIC ALGORITHM - HIPCrypto, patented in the USA under the no. US05214703, that makes use of a seven stages pipeline to be implemented as a synchronous circuit, that will be referred as micro-pipeline, coupled to an output stage as described in figure 2; so that each stage of the pipeline supplies partial results for the following stage and receives partial results from the